Berkala Fisika *ISSN : 1410 - 9662*
Vol. 7, No. 2, April 2004, hal 55 – 61

Kajian Bahasa Deskripsi Perangkat Keras

Catur Edi Widodo
Laboratorium Instrumentasi & Elektronika Jurusan Fisika FMIPA Undip

Abstrak

Telah dilakukan kajian terhadap empat bahasa deskripsi perangkat keras yaitu CDL (Computer Description Language), RTL (Register Transfer Language), VHDL (VHSIC Hardware Description Language), dan AHPL (A Hardware Programming Language). Keempat bahasa tersebut dipilih untuk dikaji karena merupakan bahasa deskripsi perangkat keras tingkat aliran data Hasil kajian menunjukkan bahwa ke empat bahasa tersebut merupakan bahasa tingkat tinggi, procedural, sesuai sistem yang dideskripsikan dan menggunakan symbol-simbol yang umum sebagai bahasa pemrogramman.

PENDAHULUAN

Proses perancangan rangkaian digital mulai dari deskripsi fungsional sampai deskripsi level gerbang disebut sintesa. Sintesa rangkaian sekuensial yang umum digunakan adalah sintesa rangkaian sekuensial dengan tabel Prosedur sintesa rangkaian sekuensial dengan tabel adalah penyusunan deskripsi fungsional, diagram keadaan, pembuatan tabel keadaan, minimisasi tabel keadaan, tabel transisi, persamaan logika dari tabel tansisi, dan di akhiri dengan deskripsi gerbang logika (Hill, 1993). Cara ini akan menjadi rumit untuk rangkaian yang besar, maka digunakan cara lain yang disebut sintesa rangkaian sekuensial dengan bahasa deskripsi perangkat keras (Davidson, 1989).

Bahasa deskripsi P/L harus dapat berkomunikasi dengan pengguna secara efisien. Bahasa pemrograman harus mudah dipahami oleh manusia. Program yang mudah dipahami akan mudah dimodifikasi dan dipelihara. Program akan mudah dimodifikasi jika mudah ditulis dan mudah dibaca. Kemudahan penulisan menjamin pemrogram dapat menuliskan program dengan mudah, sehingga pemrogram dapat berkonsentrasi pada pemecahan masalah, bukan pada penulisan program. Mudah ditulis berarti sederhana, padat, dan seragam. Sederhana berarti mudah diingat karena menggunakan istilah-istilah yang umum. Padat berarti dapat menampung seluruh ekspresi yang diperlukan dengan ringkas. Seragam berarti simbol-simbol yang dipakai tidak jauh berbeda apalagi bertentangan dengan simbol-simbol bahasa pemrograman yang lain.

Bahasa pemrograman diusahakan untuk efisien di sisi mesin dan di sisi manusia (pengguna). Efisien di sisi mesin berarti program menghasilkan kode yang efisien, sehingga penggunaaan memori sedikit dan waktu proses cepat. Bahasa akan efisien di sisi pengguna bila mudah dibaca dan mudah ditulis.

Dalam tulisan ini akan dikaji secara singkat tentang empat bahasa deskripsi perangkat keras yaitu CDL *(Computer Description Language)*, RTL (*Register Transfer Language)*, VHDL (*VHSIC Hardware Description Language)*, dan AHPL *(A Hardware Programming Language).* Keempat bahasa tersebut akan dikaji apakah

memiliki sifat-sifat seperti yang telah disebutkan diatas.

TEORI

Klasifikasi Bahasa Pemrograman

Sebuah bahasa pemrograman dapat diklasifikasikan menurut tingkat ketergantungan terhadap konfigurasi perangkat keras, tingkat kemudahan menangani tipe data, kemampuan menyelesaikan problem, tingkat kemampuan bahasa untuk dikembang-kan, tingkat interaksi program dengan peralatan, dan paradigma pemrograman yang didukung, dan. Berdasarkan klasifikasi diatas, bahasa pemrograman dapat berupa bahasa Tingkat rendah atau tingkat tinggi, prosedural atau deklaratif , orientasi ke permasalahan umum atau khusus, imperatif atau fungsional.

Bahasa pemrograman tingkat rendah berorientasi ke mesin dengan mengikuti instruksi-instruksi yang sudah tersedia dalam komputer. Bahasa tingkat tinggi biasanya berorientasi pada permasalahan, memungkinkan pemro-gram untuk melakukan abstraksi pada permasalahan. Bahasa tingkat tinggi dapat mengekspresikan program yang sama dengan bahasa tingkat rendah tanpa rincian sehingga penulisan lebih sedikit. Sebuah bahasa tingkat tinggi memberikan algoritma yang lebih alamiah. Bahasa tingkat tinggi FORTRAN dapat disebut relatif berorientasi mesin jika dibandingkan bahasa fungsional seperti LISP atau bahasa yang berorientasi obyek seperti Smalltalk.

Perbedaan atara prosedural dengan deklaratif adalah perbedaan dalam memperlakukan data. Prosedural menyebutkan urutan 'bagaimana' per-lakuan terhadap data secara terperinci. Deklaratif hanya menyebut-kan 'apa' yang dilakukan terhadap data. Tidak ada bahasa tingkat tinggi yang benar-benar deklratf atau prosedural. Bahasa tingkat tinggi selalu berada antara prosedural dan deklaratif. Contohnya, LISP dan bahasa fungsional lainya kurang prosedural dibandingkan dengan FORTRAN, tetapi lebih prosedural dibandingkan dengan PROLOG atau smalltalk.

Sebuah bahasa biasanya menangani masalah atau aplikasi tertentu lebih efisien daripada masalah atau aplikasi yang lain. Contohnya, FORTRAN adalah bahasa untuk segala keperluan (*general purpose*) yang baik untuk menangani masalah sains, teknik, dan matematik. Masalah ini memerlukan fasilitas yang lengkap untuk manipulasi data numerik. COBOL adalah bahasa untuk segala keperluan (*general-purpose*) yang baik untuk masalah bisnis dan pengolahan data. Masalah ini memerlukan fasilitas input/output yang besar, kemudahan akses ke media penyimpan tetap, dan fasilitas pelaporan yang baik.

Paradigma pemrograman adalah cara pendekatan penyelesaian suatu masalah. Saat ini paradigma pemrograman yang umum adalah paradigma imperatif. Pada paradigma imperatif program berupa urutan perintah rinci yang menginstruksikan secara pasti bagaimana masalah diselesaikan. Contoh pendekatan lain adalah paradigma fungsional yang didukung oleh LISP.

Sasaran Bahasa Pemrograman

Perancangan bahasa pemro-graman harus memperhatikan sasaran-sasaran bahasa pemrograman. Beberapa sasaran yang penting adalah komunikasi dengan pengguna, deteksi kesalahan, kemudahan penggunaan, efisiensi, dan kesederhanaan.

1. Komunikasi dengan pengguna

   Bahasa pemrograman harus dapat berkomunikasi dengan pengguna secara efisien. Bahasa pemrograman harus mudah dipahami oleh manusia. Program yang mudah dipahami akan

mudah dimodifikasi dan dipelihara. Program akan mudah dimodifikasi jika mudah ditulis dan mudah dibaca. Kemudahan penulisan menjamin pemrogram dapat menuliskan program dengan mudah, sehingga pemrogram dapat berkonsentrasi pada pemecahan masalah, bukan pada penulisan program. Mudah ditulis berarti sederhana, padat, dan seragam. Sederhana berarti mudah diingat karena menggunakan istilah-istilah yang umum. Padat berarti dapat menampung seluruh ekspresi yang diperlukan dengan ringkas. Seragam berarti simbol-simbol yang dipakai tidak jauh berbeda apalagi bertentangan dengan simbol-simbol bahasa pemrograman yang lain.

Kemudahan pembacaan berarti pemakai dengan mudah mengikuti logika program, supaya program mudah dibaca dan dipelihara. Sebagai contoh adalah sebaris program dibawah ini :

[1] -> (x) / (3)

yang mempunyai arti *jika x terpenuhi maka selanjutnya ke step 3, jika x tidak terpenuhi maka selanjutnya ke step 2.*

Deskripsi diatas tidak mudah dibaca karena kasus *jika x tidak terpenuhi maka selanjutnya ke step 2* tidak dinyatakan secara eksplisit. Pernyataan yang mudah dibaca untuk contoh diatas adalah

[1] -> (x , x') / (3 , 2) .

2. Deteksi kesalahan

Adalah suatu kenyataan bahwa pemrogram selalu membuat kesalahan. Maka dari itu diperlukan deteksi, identifikasi dan koreksi terhadap kesalahan. Bahasa pemrograman tingkat tinggi akan mempunyai kemungkinan kesalahan lebih sedikit dibandingkan dengan bahasa assembly. Misalnya penggunaan BEGIN...END pada PASCAL akan meminimumkan kesalahan aliran kontrol. Penggunaan deklarasi variabel pada PASCAL dapat menangani kesalahan secara preventif. Jika variabel direferensi, kompiler dapat melakukan cek apakah variabelnya telah dideklarasikan lebih dahulu atau belum.

Deteksi kesalahan dapat dilakukan pada saat kompilasi atau pada saat eksekusi. Umumnya, bahasa dirancang untuk melakukan deteksi kesalahan pada saat kompilasi. Lebih mudah dan aman untuk menemukan semua kesalahan pada saat kompilasi daripada pada saat eksekusi.

3. Kemudahan penggunaan

Bahasa pemrograman dirancang agar mudah untuk dipelajari dan diingat. Pemrogram tidak harus selalu melihat manual. Bahasa harus sederhana dan langsung mengenai sasaran. Tidak banyak cara untuk mengekspresikan program.

4. Efisiensi

Bahasa pemrograman diusahakan untuk efisien di sisi mesin dan di sisi manusia (pengguna). Efisien di sisi mesin berarti program menghasilkan kode yang efisien, sehingga penggunaaan memori sedikit dan waktu proses cepat. Bahasa akan efisien di sisi pengguna bila mudah dibaca dan mudah ditulis.

5. Kesederhanaan

Kesederhanaan dapat dicapai dengan pembatasan obyektif (tujuan), penjagaan terhadap kemudahan pembacaan, dan berdasar pada kesederhanaan konsep. Perlu diingat bahwa kesederhanaan tidak dapat dicapai dengan struktur yang jelek, karena akan mengakibatkan kekacauan. Kesederhanaan juga tidak dapat dicapai dengan cara pembatasan hal-hal yang sudah umum (generalitas), karena akan

berakibat implementasi tidak komplit.

METODOLOGI

Metode yang digunakan dalam kajian ini adalah:

1. Penelusuran pustaka: mengamati bermacam-macam jenis bahasa deskripsi perangkat keras, terutama klasifikasi dan sasaran bahasa deskripsi perangkat keras.
2. Pemilihan sampel program yang lengkap: pada tiap tiap bahasa yang dipilih, dipelajari sebuah program yang algoritmanya cukup lengkap. Lengkap artinya memiliki perintah input data, transfer data, aliran data, pengambilan keputusan, operasi logika, dan output data.
3. Anallisis sampel program terhadap kriteria bahasa pemrograman: tiap sample program yang dipilih dianalisis berdasarkan kemudahan dipahami, kemudahan penulisan, efisiensi, dan keseragaman.
4. Penarikan kesimpulan

HASIL DAN PEMBAHASAN

Computer Description Language (CDL).

CDL (Baer, 1980) berupa sekuen statemen deklarasi dan eksekusi. Deklarasi digunakan untuk meggambarkan penyimpan (storage) yaitu register atau memori. Perintah eksekusi digunakan sebagai kontrol stuktur memori dan register tersebut.

Contoh deskripsi dengan CDL

```
01    Register   MAR (0-15)
02               MDR (0-31)
03               LC (0-15)

04               IR (0-31)
05               INTRP
06    Memory     M(MAR) = M(0-13172 , 0-31)
07
08    /STARTOFFETCH/
09                  IF (INTRP <>0) THEN GOTO INTERSEQ;
10                  MAR <- LC;
11                  MDR <- M(MAR), LC <- countup LC;
12                  GOTO STARTOFFETCH;
13    /INTERSEQ/
14                  Interrupt sequence
15                        .
16                        .
17                  GOTO STARTOFFETCH;
```

Deskripsi diatas adalah *fetch cycle* yaitu proses mengambil kode perintah dari memori. Baris 1 sampai 6 mendeskripsikan register dan memori yang dipakai. Baris 8 sampai 17 mendeskripsikan eksekusi dari register dan memori yang dideskripsikan tersebut. Mula-mula isi register LC ditransfer ke register MAR, kemudian isi memori yang alamatnya ada pada register MAR ditransfer ke MDR dan nilai LC di naikkan. Jika isi INTRP tidak sama dengan nol, maka dilakukan proses selanjutnya.

Deskripsi dibagi menjadi dua bagian yaitu deklarasi (baris 1-6) dan eksekusi (baris 8-17). Instruksi yang dipisahkan tanda koma (baris 11) merupakan instruksi paralel. Instruksi yang dipisahkan tanda titik koma merupakan instruksi sekuensial. Label digunakan untuk keadaan pilihan GOTO.

Struktur program (yang berupa deklarasi dan eksekusi), pemakaian label, dan penggunaan notasi transfer "<-", sesuai dengan sasaran komunikasi dengan pengguna, tetapi disini tidak ada koneksi. Transisi dapat dilakukan dengan GOTO, tetapi struktur GOTO (baris 12 dan 17) tidak sesuai dengan sistem sekuensial.

Register Transfer Language (RTL).

RTL (Mano, 1994) digunakan untuk menggambarkan perpindahan bit antar register dalam mikrooperasi. Statemen-statemen pada RTL berupa fungsi kontrol dan mikrooperasi. Mikrooperasi berupa register transfer, aritmatik, logika, dan pergeseran.

Contoh deskripsi dengan RTL

```
01      T0:     AR <- PC
02      T1:     IR <- M[AR], PC <- PC+1
03      T2:     D0,...,D7 <- Decode IR(12-14),
04                AR <- IR(0-11),  I <- IR(15)
05      D0T3:   DR <- M[AR]
06      D0T4:   AC <- AC ^ DR, SC <-0
07      D1T3:   DR <- M[AR]
08      D1T4:   AC <- DR, SC <-0
09                    .
10                    .
11      D7T3:  DR <- M[AR]
12      D7T4:  DR <- DR+1
13      D7T5:  M[AR] <- DR, if (DR=0) then (PC <- PC+1), SC <- 0
```

Deskripsi diatas adalah instruksi *fetch cycle* (baris 01 sampai 04) yang diikuti AND (baris 05 dan 06), ALD (baris 07 dan 08), dan ISZ (ISZ = is zero, baris 11 sampai 13). Pada *fetch cycle* mula-mula isi PC (program counter) ditransfer ke AR (address register). Kemudian isi memori yang ditunjukkan register AR di transfer ke register IR. Isi register AR dari 0 sampai 11 ditransfer ke register AR, isi register AR bit terakhir (bit 15) ditransfer ke register bendera I, dan isi register AR dari 12 sampai 14 di-kode-kan menghasilkan D0,D1..D7. Jika yang bernilai satu adalah D0, maka instruksi selanjutnya adalah AND, jika yang bernilai satu adalah D1, maka instruksi selanjutnya adalah LDA, dan seterusnya.

Instruksi terdiri dari kontrol dan time (misalnya D7T3:) , dan register transfer, (misalnya DR <- MAR). Instruksi dalam satu kontrol dan time merupakan instruksi paralel. Dengan penggunaan time secara eksplisit, maka tidak ada perintah GOTO, karena waktu selalu maju.

A Hardware Description Language (AHPL)

AHPL (Hill,1993) digunakan untuk mendeskripsikan urutan operasi dan aliran informasi dari satu titik ke titik lain dalam komputer. Deskripsi dalam AHPL disebut module yang berisi

nama, deklarasi, kontrol sekuen, dan terminasi.

Contoh deskripsi dengan AHPL :

```
01 MODULE :   SEQUENTIAL COMPARATOR
02 INPUTS :   X[8]; a ; b .
03 MEMORY :   A[8] ; COUNT[2] .
04 OUTPUTS :  Z[8] ; out .
05 CLUNITS :   GREATHER <: GREATHERTHAN(X,Y)
06
07     1  -> (~a) / (1).
08     2  A <- X .
09     3  (~b) / (3) .
10     4  A * GREATHER(X,A) <- X; COUNT <- 0,0.
11     5  COUNT <- COUNT[0] + COUNT[1], ~COUNT[1] ;
12        out = 1;
13        -> (^/COUNT , ~^/COUNT) / (1,5) .
14     ENDSEQUENCE
15     CONTROLRESET(1); Z=A.
16     END.
```

Deskripsi diatas adalah sebuah komparator sekuensial 8 bit antara input X dengan input X sebelumnya. Untuk memasukkan nilai X yang pertama (baris 8), input a harus bernilai satu (baris 7). Untuk memasukkan nilai X yang kedua input b harus bernilai satu. Langkah selanjutnya adalah membandingkan X yang pertama (yang telah disimpan di A) dengan X yang kedua. Jika X lebih besar dari A, maka nilai X ditransfer ke A (baris 10, A*GREATHER(X,A) <-X.). langkah ke 5 (baris 11 sampai 13) menunggu tiga kali periode untuk kembali ke langkah 1.

Sekelompok instruksi dipisahkan dengan kelompok yang lain dan diberi nomor urut yang disebut step. Sekelompok instruksi dalam satu step dikerjakan secara paralel. Secara normal, instruksi akan dikerjakan dari step 1, 2, 3 dan seterusnya. Jika instruksi selanjutnya bukan nomor yang urut, digunakan instruksi percabangan, contohnya, baris 07 berarti IF (a=0) THEN GOTO 1 ELSE GOTO 2

Selain pada percabangan, AHPL cukup memenuhi sasaran komunikasi dengan pengguna, kesederhanaan, dan kemudahan penggunaan. Instruksi percabangan tidak memenuhi kemudahan pembacaan. Dengan banyaknya kriteria yang terpenuhi, notasi AHPL akan dipakai bahasa D.

VHSIC Hardware Description Language (VHDL)

VHDL berguna untuk mendeskripsikan komponen hardware dan perilaku kerja komponen tersebut. Terdiri dari spesifikasi antar muka yang disebut entity dan spesifikasi perilaku yang disebut architecture.

Contoh deskripsi dengan VHDL :

```
01      ENTITY halfadder
02      IS
03         PORT (x,y,cin:IN BIT;
04               cout,sum:OUT BIT)
05      END
06
07      ARCHITECTURE dataflow OF halfadder
08      IS
09         SIGNAL c:BIT;
10      BEGIN
11         sum <= x XOR y XOR cin AFTER 5 NS ;
12         c <= (y AND cin) OR (x AND cin) OR (x AND y);
13         cout <= c AFTER 6 NS;
14      END.
```

Deskripsi diatas adalah sebuah half adder dengan masukan x,y, dan cin (*carry input*) dan keluaran sum dan cout (*carry output*). Port input-output didelarasikan dalam entity, dan perilakunya dideklarasikan dalam architecture. Sum diperoleh dari operasi XOR terhadap x, y, dan cin. Carry diperoleh dari operasi OR terhadap y AND cin, x AND cin, dan x AND y.

Semua perintah pada bagian architecture adalah paralel. Untuk proses sekuensial, digunakan perintah-perintah sekuensial yaitu IF, CASE, LOOP, ASSERT, dan WAIT.

VHDL memiliki struktur program yang jelas, yaitu entity dan architecture. Entity merupakan antar muka, dan architecture merupakan perilaku entity tersebut.

KESIMPULAN

1. CDL cukup bagus dalam komunikasi dengan pengguna, tetapi struktur GOTO tidak sesuai dengan sistem digital sekuensial sehingga untuk sintesa, bahasa ini kurang cocok.
2. RTL cocok untuk mikro-operasi, tetapi tidak cocok sintesa rangkaian digital sekuensial karena tidak ada transisi yang eksplisit.
3. AHPL cukup baik dalam komunikasi dengan pengguna, kesederhanaan, dan kemudahan penggunaan.
4. VHDL memiliki struktur program yang jelas, yaitu entity dan architecture. Entity merupakan antar muka, dan architecture merupakan perilaku entity tersebut. VHDL cukup baik dalam komunikasi dengan pengguna tetapi paling rumit dibandingkan degan bahasa deskripsi yang lain.

DAFTAR PUSTAKA

Hill, F.J., “Computer Aided Design with Emphasison on VLSI”, John Willey and Sons, 1993

Narali Z, “VHDL Analysis and Modeling Digital System:, Mc Graw-Hill, 1993

Mano,M.M., “Digital Logical and Computer Design”, Prentice Hall Inc, 1994

Lee, S.C, “Modern Swiching Theory and Digital Design” (terjemahan), Penrbit Erlangga, 1978